SeDiCom: A Secure Distributed Privacy-Preserving Communication Platform

Alexander Marsalek[1,2(✉)], Bernd Prünster[2], Bojan Suzic[2], and Thomas Zefferer[3]

[1] Secure Information Technology Center Austria, Graz, Austria
Alexander.Marsalek@iaik.tugraz.at
[2] Graz University of Technology, IAIK, Graz, Austria
{BPruenster,BSuzic}@iaik.tugraz.at
[3] A-SIT Plus GmbH, Vienna, Austria
Thomas.Zefferer@a-sit.at

Abstract. Efficient and secure electronic communication is crucial for successful business-to-business processes. Due to the weaknesses of e-mail communication, a shift towards instant messaging can also be observed in this context. However, reliance on instant-messaging solutions in business processes has its own drawbacks such as the lack of archiving capabilities and unsatisfactory legal compliance. Furthermore, special business scenarios such as bidding processes come with complex security requirements that are not met by current instant-messaging solutions. To also enable efficient and secure electronic communication for these scenarios, we propose a blockchain-based instant-messaging solution under the name SeDiCom. SeDiCom employs the capabilities of the blockchain technology, one-time identities, and the Tor anonymity network to enable confidential instant messaging without leaking any identifying metadata. Our proposed solution provides non-repudiation, censorship resistance, integrated backup facilities, and verifiable notices of receipt, while inherently preventing man-in-the-middle attacks and virtually all other forms of eavesdropping. By this means, SeDiCom enables efficient and secure electronic communication for business scenarios with special security requirements while also catering to today's usage patterns.

Keywords: Blockchain · Messenger · Decentralized
Secure data exchange · Censorship-resistant · Non-repudiation
Privacy preserving

1 Introduction

During the past decades, many workflows and a significant amount of written communication have been migrated from pen-and-paper procedures to e-mail. While person-to-person communication is today typically carried out using

V. Ganapathy et al. (Eds.): ICISS 2018, LNCS 11281, pp. 345–363, 2018.
https://doi.org/10.1007/978-3-030-05171-6_18

instant messengers, with *WhatsApp* having reached over a billion active users [1], business workflows still rely heavily on e-mail. SMTP, the protocol e-mail delivery is based on, was, never designed with confidentiality and authenticity in mind, however, even though it is still heavily used to transport confidential information. In the more recent past, however, this has also begun to change in favor of instant messaging systems. Popular platforms like *Apple*'s *iMessage*[1], *WhatsApp*[2] and *Viber*[3] have recognized the demand for secure and confidential communication and provide (end-to-end) encrypted communication [2–4]. Recently, the need to integrate such messaging solutions into business processes has also been identified [5–7].

Relying on instant messaging in a business context instead of e-mail has its own drawbacks, such as the lack of archiving facilities and increased difficulties when it comes to legal compliance [8,9]. Moreover, certain business processes are subject to specific and complex constraints: Aside from archiving, communication between parties must be kept secret (including metadata) while it happens. At a later point in time, however, any of the involved parties may need or wish to prove to the public not only that a communication took place but also when and about what. Examples include certain bidding processes or tender offers. In such cases, none of the parties involved must be able to repudiate or refute any past statements. More than end-to-end encrypted messaging is required to reach these goals, as encrypting traffic does not prevent third parties from learning about users' communication habits [3,10]. As long as detailed, unaltered metadata of electronic conversations is available, profound conclusions about communicating parties and even the contents of a conversation can be drawn, which enables profiling [11,12] and can serve as a basis for critical decisions [13].

Although increasingly more solutions for encrypted online communication exist [2,14,15], virtually all of them still leak metadata [3]. The factors contributing to this drawback can be traced back to the underlying architecture, centralized designs and the reliance on a single authority. Furthermore, the mechanisms used for key management and the reliance on stable identities of the parties involved may affect the security and privacy of users in numerous ways. In this work, we propose a blockchain-based [16] communication service which not only offers end-to-end encryption, but also guarantees untraceability of all communication. Moreover, our approach inherently supports unforgeable identities, virtually eliminating the risk of man-in-the-middle attacks. Users can therefore be sure about whoever they are communicating with at any time. Our platform also proves whether a particular message has been delivered and read. By relying on the blockchain technology, our design also guarantees immutability of all message contents, non-repudiation and censorship resistance. Essentially, only the parties involved in a conversation know who communicates with whom and about what, moreover nobody can later dispute any statements previously made. Our design is therefore well-suited for transmitting any form of confidential infor-

[1] https://support.apple.com/explore/messages.
[2] https://www.whatsapp.com/.
[3] https://www.viber.com/.

mation which might require a notice of receipt and non-repudiation, such as registered electronic mail or legal documents like contracts, for example. While our proposed solution has several advantages, it also comes with some limitations rooted in its blockchain underpinnings such as every participant receiving all (encrypted) messages. However for our envisioned use-cases these drawbacks do not pose an issue.

Our novel approach of relying on one-time addresses, a well-defined address schedule and incorporating the *Tor* [17] anonymity network prevents any and all leaks of conversation metadata. The only detail observable from outside a conversation is the fact that someone is sending something into a blockchain-based network. Due to the nature of blockchain-based systems, it is not even possible to deduce whether someone is receiving any messages at all. At the same time, the most prominent scalability issues associated with blockchain-based designs do not apply, as explained in Sect. 3. To the best of our knowledge, no existing system offers such extensive privacy-preserving mechanisms.

The remainder of this paper is structured as follows. In the next section, we discuss related work and provide the necessary background information. Section 3 presents the architecture of SeDiCom in detail, followed by a performance evaluation in Sect. 4 and an extensive security evaluation in Sect. 5. Finally, we conclude this paper in Sect. 6 and provide an outlook.

2 Background and Related Work

Since we are presenting a blockchain-based communication platform providing an extensive set of security and privacy properties, we first provide some basic background on distributed ledger technologies commonly referred to as blockchain. The blockchain was developed by someone under the pseudonym Satoshi Nakamoto as the decentralized ledger for the cryptocurrency Bitcoin [16] and consists of a list of cryptographically linked blocks. Each block includes the fingerprint of the previous block and a list of transactions. Consensus algorithms are used to agree on which chain of blocks is considered valid. Bitcoin uses a proof-of-work (PoW) consensus algorithm: Network nodes, which want to create new blocks, so-called *miners*, have to solve a cryptographic task (mining). The first one solving the task can create a new block and receives a specified quantity of Bitcoin units as a reward. An attacker willing to delete or modify a block has to redo all the work done by the remaining (honest) network as the network always uses the chain with the highest combined PoW. Later incarnations of blockchain-based systems like *Ethereum* (testnet) [18] do not require computationally intensive tasks to process transactions, but rely on other concepts like proof-of-authority (PoA) as implemented by the *Clique* [19] algorithm. Regardless of the underlying consensus mechanism, a blockchain provides a tamper-proof, trackable, fault-tolerant, DDoS-resistant, censorship-resistant, distributed public ledger, as long as more than half the network's computation power[4] is controlled by honest nodes.

[4] In case of a proof-of-work based blockchain.

The advantages of blockchain-based data storage combined with encrypted communication are employed by several privacy-preserving messaging platforms. Systems like the *VooMessenger*[5] or closed-source products like *Echo*[6], *Blokcom*[7] by *Reply S.p.A.*[8] and *CrypViser* [15] are examples for such platforms.

However, all of the above described properties, which are a blockchain-based system's greatest strength can also turn into its greatest weakness from a privacy point of view: When using a blockchain-based system to exchange confidential information, it presents various short-term advantages like tamper-resistance and fault-tolerance. In the long run, however, this can backfire. Considering that all messages ever transmitted using such a system are persistently stored inside a single highly replicated database, which is accessible from anywhere on the Internet, a single flaw in the encryption mechanism can lead to massive data breaches in the future. From this point of view, the utility of blockchain-based messaging platforms for increasing privacy has to be classified as questionable. However, some use cases remain where this is not an issue.

Our system caters to such scenarios where confidentiality (of contents and meta data) is paramount at first, but communication contents are usually divulged at a later point in time. We achieve this by building on the solid foundation of the blockchain and augmenting it to provide extensive guarantees wrt. privacy, non-repudiation, integrity, and availability.

More traditional messaging systems, but also various other blockchain-based messengers, are inherently incapable of achieving these goals due to their architecture [2,14,15].

In the next section we will introduce our proposed solution.

3 SeDiCom

We propose a messaging platform which enables secure and private communication over the Internet without leaking metadata. After all, metadata can be sufficient to severely endanger users' privacy [10–13,20]. Based on the properties of typical messaging solutions and our goals, we can derived the following requirements:

R1 Confidentiality: Only entitled entities may read messages.
R2 Message Authentication: Nobody may alter a message or forge a sender's identity (contrary to what is possible with e-mail addresses [21,22]).
R3 Metadata Protection: It must not be possible to obtain the identities of the communicating parties. At the same time, a user must be able to produce a proof of sending or receiving a message if she chooses to.
R4 Decentralization: There shall be no central instance that can be attacked to obtain unencrypted messages or cryptographic material to decrypt messages. No instance shall be able to collect user-related data, like messages, or metadata.

[5] https://faizod.com/blockchain-solutions/business/voomessenger/.
[6] https://my-echo.com/.
[7] http://www.reply.com/en/content/blokcom.
[8] https://reply.com/.

R5 Proof of Existence: The sending time of a message must be provable.
R6 Notice of Receipt: Notices of receipt shall be produced and transmitted automatically entailing a proof of existence (POE).
R7 High Availability, Redundancy: Messages have to be replicated to gain a failure-resistant system. The system should be resistant to denial-of-service (DoS) attacks.
R8 Non-Repudiation: Authors shall not be able to successfully challenge the authorship of a message.
R9 Immutability: Once a message has been sent, it must not be possible to manipulate (e.g. modify or remove) it in any way.
R10 Censorship Resistance: It shall not be possible to block individuals from using the service.

Notably, scalability is not listed as a requirement. The reasons for this apparent lack of an elementary requirement for applications as targeted by our design are threefold: First, current-generation blockchain designs do not suffer from as many scalability issues as Bitcoin. Second, the targeted use case of our design imposes fewer and weaker constraints compared to WhatsApp, for example: Bidding processes and tender offers typically do not require the exchange of millions of messages in a short time frame and the number of participants is typically also lower. Finally, our work focuses on security and privacy and achieves its goals by building upon guarantees provided by blockchain designs (as described in the following section). Advancements made wrt. blockchain scalability thus directly benefits our solution. We therefore expect performance to improve over time.

Our solution proposed in this section meets all identified requirements. We discuss the architecture of this solution in the following section.

3.1 Architecture

Considering the requirements identified in Sect. 3, we have derived a suitable architecture for our **se**cure **di**stributed **com**munication platform, called SeDiCom. We use a blockchain [16] as the central component. The blockchain provides a highly available, highly redundant, censorship-resistant network communication and storage system, the content of which cannot easily be modified or erased. Figure 1 shows the high-level architecture of our proposed solution. It shows an arbitrary number of clients (nodes) running our software. These clients are connected via the Tor network (visualized as onions) and create a P2P network that maintains and distributes a blockchain. We use transactions to transfer information in this P2P network. These transactions are broadcast through the network. Each transaction contains at least one sender address (input) and one receiver address (output) and is thus ideal for sending content to one or more participants. As every participant can generate addresses on their own and there is no binding to an identity, users remain anonymous. Once a transaction is mined into a block, which is accepted by the network, it can be considered immutable. Therefore, the blockchain provides a highly available, decentralized

backup system. The creation time of a block can be used as POE for all included transactions. The sender and receiver addresses are representations of the corresponding public part of the public-private key-pairs. By using the public key of the collocutor, it is possible to encrypt messages. These messages can only be decrypted by the owner of the corresponding private key. Using the private key, it is possible to sign messages and therefore prove key ownership and implicitly prove identity. For the rest of this paper, we will refer to these public-private key pairs as identities. We propose a protocol, where every participant creates a main identity and several one-time identities. The main identity is only used for identity proofs. It will never be used to send or receive messages. Instead, one-time identities are used to send and receive messages. In the next section, we describe a typical process flow between two collocutors.

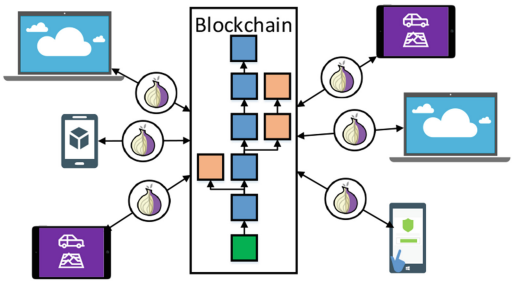

Fig. 1. Proposed solution architecture.

3.2 Generic Process Flow

In this section, we describe a typical process flow between two collocutors who want to securely communicate with each other. The sequence of actions is visualized in Figure 2 for two users, namely Alice and Bob. Initially, both parties create their main identity ID_{AB} and ID_{BA} (1-2). Every identity ID_{XY} consists of a private key ID^{-}_{XY} and a public key ID^{+}_{XY}. X denotes the owner of the key, and Y the collocutor. Knowing the public key is equivalent to knowing the corresponding address. Depending on their needs, participants can choose to create several main identities, e.g., one per collocutor. For the protocol it is only important that the same main identity is used for the same collocutor. Otherwise, it would not be possible to verify the message sender's authenticity. The main identity is never used to send or receive transactions, it will only be used to sign messages encoded into transactions. Next, both parties create a one-time send ($IDES_{AB1}$, $IDES_{BA1}$) and receive identity ($IDER_{AB1}$, $IDER_{BA1}$) (3-4) for sending and receiving transactions. These identities will be created for every collocutor individually. Next, the public keys of the main identities ID^{+}_{AB}, ID^{+}_{BA} and the public key (address) of the receiving identity of Bob $IDER^{+}_{BA1}$

have to be exchanged over an authenticated channel (5), e.g. by digitally signing them using a qualified electronic signature and exchanging the result electronically or by meeting somewhere in person. $IDER$ denotes a one-time, ephemeral identity that is used to receive messages, whereas $IDES$ represents a one-time identity that is used to send messages. The information exchanged in Step 5 is not secret, it is only used to authenticate the other party and to communicate details on how to reach the other party. After the exchange, Alice has all necessary information to communicate securely and anonymously with Bob. Next, Alice sends a message m to Bob and prepends her receiving address $IDER^{+}_{AB1}$ to the message. The resulting message is signed using Alice's main identity key ID^{-}_{AB} to prove that the message was indeed sent by her and then signed with $IDES^{-}_{AB1}$, to prevent replay attacks. From now on we refer to the first signature as inner signature and to the second signature as outer signature. The resulting message is shown in Eq. 1. Note that $Sign(m)$ returns m and the signature over the fingerprint of m.

$$m_{Signed} = \text{Sign}_{IDES^{-}_{AB1}}(\text{Sign}_{ID^{-}_{AB}}(IDER^{+}_{AB1}||m)) \tag{1}$$

As the message will be sent over the P2P-network, the next step is to encrypt the message, thus preventing unauthorized parties from eavesdropping on it. Next, a shared key is derived using the Diffie–Hellmann key agreement protocol. Both

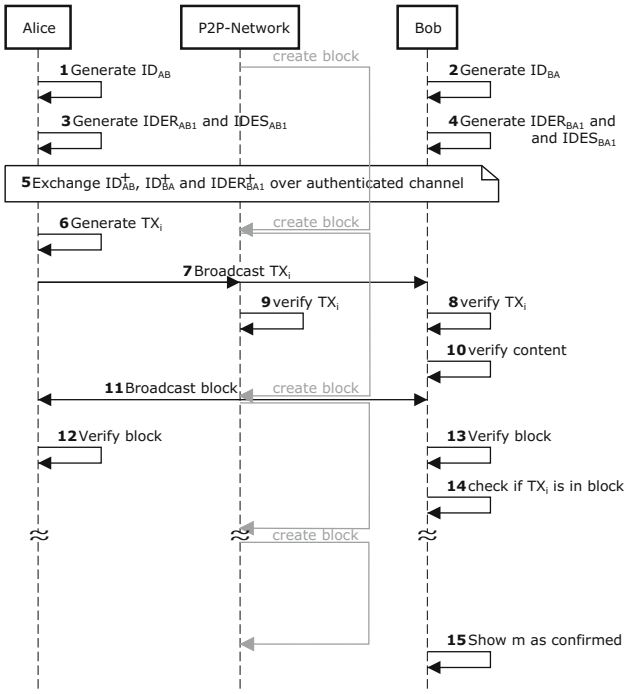

Fig. 2. Exchange of the first message between two parties.

parties can derive the same shared key using their private key and the public key of the collocutor as shown in Eq. 2.

$$\begin{aligned} SharedSecret &= \text{ECDH}(IDES^{-}_{AB1}, IDER^{+}_{BA1}) \\ &= \text{ECDH}(IDER^{-}_{BA1}, IDES^{+}_{AB1}) \end{aligned} \tag{2}$$

Next, the shared secret is hashed using SHA-256 and the resulting fingerprint is used as key for an symmetric cipher as shown in Eqs. 3 and 4.

$$K_{Msg} = \text{SHA-256}(SharedSecret) \tag{3}$$

$$C_{Msg} = \text{E}_{K_{Msg},IV}(m_{Signed}) \tag{4}$$

Note that neither of the parties has to store the key K_{Msg}. Next, the encrypted message C_{Msg} together with information needed to decrypt the message (depending on the cipher and the chaining mode; e.g. an IV) are packed into a container C as shown in Eq. 5.

$$C = IV||C_{Msg} \tag{5}$$

Next, a new transaction is created, with $IDES^{+}_{AB1}$ as sender and $IDER^{+}_{BA1}$ as receiver. This container C is encoded into the transaction's data field. Finally, the transaction will be signed with $IDES^{-}_{AB1}$ to gain a valid transaction TX_i (6) and broadcast to the network (7). As the transaction is sent from a one-time identity to another one-time identity, no conclusions can be drawn about who communicates with whom. Shortly after sending a message, Bob and other nodes in the network will receive the transaction TX_i and verify it (8-9). Mining nodes will save valid transactions and include them into the next block. Bob will check if the transaction is addressed to him by checking if the receiver address of the transaction matches one of his active ones. If it matches, Bob will decrypt the content using his corresponding private key $IDER^{-}_{BA1}$ and subsequently verify the outer signature. If the outer signature matches the sender's identity of the transaction, Bob continues to verify the inner signature.

As every receiving address is created for a specific collocutor, Bob can verify whether the signature matches the expected sender's identity. If one of the signatures does not match the expected identity, Bob aborts the process. If the inner signature confirms the expected identity, Bob separates the content m and the reply address $IDER^{+}_{AB1}$ (10). The reply address $IDER^{+}_{AB1}$ will be stored for later use and the content m will be shown as *unconfirmed message*. Unconfirmed means that the message is not yet included in a block, thus not all properties (like high availability, for example) hold yet. After the transaction is included in a block by a miner, the block will be broadcast to all network participants (11). Every participant will verify whether the block is valid (12-13). If the block is valid, meaning all transactions are valid and it references the previous valid block, Bob verifies whether TX_i is included (14). To be sure, he will wait until the block's maturity depth[9] is equal or greater than a selected threshold. If the

[9] As an example, Bitcoin wallets typically require a maturity depth of six blocks. Meaning six additional blocks need to be created after the block containing the corresponding transaction.

block's maturity depth requirement is fulfilled, the message will be shown to the user as *confirmed message* (15).

If Bob wants to reply to Alice, he generates a new one-time send and receive identity ($IDES_{BA2}$ and $IDER_{BA2}$) and repeats Steps 6 and 7 with opposite roles. Bob will use the saved reply address, extracted from TX_i as target address for his generated transaction. This reply message is simultaneously a proof for Alice that Bob has received the previous message. If one party wants to send another message before a reply is received from the other party, the last receiving identity of the collocutor is used again, but a new sending and receiving identity is created each time, to prevent anyone from concluding anything from it. The new receiving identities are necessary for the read confirmation requirement. Once the other party answers, she will reply to the last received reply address and append all other unused reply addresses to the message. One's one-time identities do not need to be remembered, they can always be derived from the main identity. It is only necessary to remember the number of already used one-time identities. For performance reasons, it makes sense to also remember the unused reply addresses submitted by each collocutor and the unused reply addresses submitted to collocutors.

3.3 Security Features

The blockchain already provides several features, like decentralization, proof of existence high availability, and redundancy. This section highlights two security features of our proposed solution, namely the address schedule used to generate one-time identities, and the proof of receipt feature.

Address Schedule. We describe the process of deriving one-time identities from the main identity. A key requirement for the derivation process is that individual one-time identities can be published without disclosing any information about other identities. We derive our one-time identities based on Eq. 6, where ID_{YZ} is the main identity of user Y used for communicating with user Z. X defines the type of identity (send or receive) to create, i is an integer that is increased every time a new one-time identity of this type is created, H(...) denotes a cryptographic hash function and genKeyPair(...) generates an EC key pair.

$$IDEX_{YZi} = \text{genKeyPair}(\text{H}(ID^{-}_{YZ} + \text{H}(i||Z||X))) \tag{6}$$

The security of one-time identities is based on the one-way characteristic (preimage resistance) of the hash function.

Proof - Message Sent/Received. If a party wants to prove that a message was sent by the other party, it needs to publish the corresponding receiving identity. Using this identity, everyone can extract the corresponding transaction from the blockchain and decrypt it. The inner signature of the decrypted content will match the main identity of the other party and thus prove that she signed it.

If Party A wants to prove that she sent a message to B, she has to publish the corresponding sending identity and the receiving identity, on which she received the reply address. Using the sending identity, everyone can extract the corresponding transaction from the blockchain and decrypt it. Thus, it can be shown, that the message was sent and signed by party A. Next, it must be proven that the address the transaction was sent to belongs to Party B. This can be done by publishing the receiving identity on which the reply address from B has been received. The corresponding decrypted message will be signed by Party B and contain the used reply address.

3.4 Implementation

We built a proof-of-concept implementation of the proposed solution based on the Ethereum [18] blockchain. Our implementation relies on Clique [19], instead of a PoW consensus algorithm to improve performance. Furthermore, we reduced the block interval to one minute, to reduce the time until a transaction is included in a block. This also increases the throughput of our solution in terms of messages per time interval[10]. These adjustments remedy virtually all scalability concerns typically associated with blockchain-based systems. We modified our Ethereum implementation, enabling sending transactions for free. Otherwise, every one-time identity would need a balance to pay the fees. This step was necessary, as we currently have no way to send a balance to one-time identities from another identity without leaving metadata, such as which identities belong together. This procedure removes one of Ethereums' main protection mechanisms against attacks that wastes the power of the network or endangers network stability, e.g., endless-loops in smart contracts. We deal with this issue by deactivating Ethereums' smart contract functionality[11]. While this prevents many attacks, spamming the network with useless, but valid, transactions is still possible. These messages will not be shown to the user, but waste space in blocks. We plan to deal with this issue in future work and describe a possible approach in Sect. 6. We implemented our prototype in Java, using the web3j[12] library as connector to a modified version of Geth[13], an Ethereum node implementation written in Go. Our Java application communicates with Geth via JSON-RPC. For simplicity, we tested our approach with a single miner. The implementation demonstrates the feasibility of the proposed solution, which is platform-independent and lightweight enough to be used on desktop systems as well as mobile systems like smartphones.

Ethereum uses the Elliptic Curve SECP-256k1 [23] to generate EC keys for transaction signing. Our proposed one-time identities also rely on this curve. As a consequence, we can use the Elliptic-curve Diffie–Hellmann (ECDH) key

[10] Ethereum does not define a maximum block size in contrast to e.g. Bitcoin.
[11] Transactions related to the creation of a smart contracts are ignored and not included into a block.
[12] https://github.com/web3j/web3j.
[13] https://geth.ethereum.org/.

agreement protocol to derive a shared key. This enables the sender and receiver of a transaction to independently calculate a shared secret based on one's private key and the collocutor's public key. Next, the shared secret is hashed using SHA-256 and the resulting fingerprint is used as the key for an AES cipher in counter mode without padding. For the cipher we generate a random initialization vector (IV), which is prepended to the encrypted message C_{Msg}. No other external information, (not available in the transaction) is needed to decrypt the message. Next, we discuss the performance of the proposed solution.

4 Performance Evaluation

This section provides a short overview of the performance impact in terms of size and speed overhead. Note that we did not consider the blockchain performance, as it is independent of our proposed protocol. Instead, we focus on the overhead in size imposed by our protocol and show that the necessary signing, verification, encryption, and decryption steps have no practical consequences in terms of performance.

4.1 Required Overhead

Figure 3 shows the size overhead of the proposed solution for different message lengths, compared to encoding the messages directly into transactions without one-time identities and without signing and encrypting them. As can be seen, the overhead is mainly noticeable for smaller input message sizes since our solution produces a constant overhead independent of the input. In detail, our proposed solution requires 42 bytes for submitting the new receiving address, 65 bytes each for the inner and outer signature, as well as 16 bytes for the IV. This results in 188 bytes overhead. Finally, the payload has to be hex-encoded, which doubles the size. To summarize, our proposed solution produces a per-message overhead of $188 \times 2 = 376$ bytes. Next we discuss the throughput.

4.2 Throughput

We measured how many messages per second can be created and how many received messages per second can be verified. We executed the performance test on a MacBook Air with an Intel Core i7-4650U CPU running at 1.7 GHz using an unoptimized single threaded Java program. The test consists of first signing and encrypting 10,000 messages and then decrypting and verifying 10,000 messages. Based on the time required we calculate the number of messages that can be created and verified every second. We repeated the experiment with different message sizes, namely 1 byte, 1024 bytes, 8192 bytes and 16384 bytes. Note: The test was performed without actually sending the messages, thus ignoring the performance of the Ethereum network. Table 1 shows the results.

The results show that our proposed solution has a small overhead and an minor impact on the performance and is thus well suited for its intended use-cases. Note that the presented solution is not intended to replace a messaging

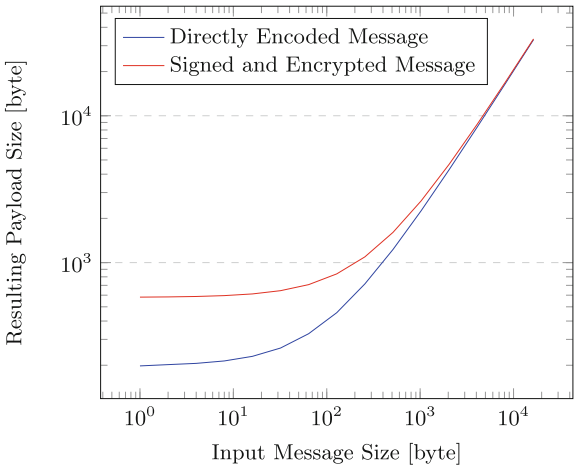

Fig. 3. Comparison of payload sizes of directly encoded messages versus messages sent by the proposed solution. Note the logarithmically scaled x- and y-axes.

Table 1. Throughput of messages per second that can be generated or read for different input sizes.

Size [byte]	Throughput create [Msg/sec]	Throughput read [Msg/sec]
1	810.4	1056.6
1024	827.5	991.4
8192	682.9	833.0
16384	587.8	729.9

platform like Whatsapp. Rather, it aims at specific, but all the more critical use-cases, which require a secret communication process that is likely to be disclosed later on, e.g., a bidding process where all bidders first send their offers without leaking any data and later on the offers can be disclosed to ensure a fair procurement.

5 Security Evaluation

To verify that our proposed approach provides the proclaimed security features, we performed a security evaluation loosely based on the *Common Criteria for Information Technology Security Evaluation* (CC)[14].

We derive security goals and assets from the requirements presented in Sect. 3. Our assumptions rely on the proven security of all underlying technologies and expected secure usage of existing systems. Similarly, the security properties applied in this analysis are derived from our model presented in Sect. 3.

[14] https://iso.org/standard/50341.html.

We further designate the scope of our analysis to the case that provides adequate protection against all but a highly determined attack that requires a tremendous amount of resources and effort.

After elaborating each of these categories in the subsequent sections, we conclude this chapter with the comprehensive overview of threat mitigation mechanisms.

5.1 Assets

A1 Message: Message contents are considered confidential. Only the interlocutors are eligible to read it, unless one involved party wants to publish it.
A2 Main Identity: The private keys belonging to the main identities must not be disclosed. The public part does not need to be protected.
A3 One-time Identities: Private keys of one-time identities are considered confidential until the owner decides to disclose them, e.g. to prove that they sent or received a message.
A4 Metadata: Metadata that can be used to conclude anything about the senders' or receivers' identity, is considered confidential.

5.2 Assumptions

AS1 Secure/Trusted Devices: We assume that users and miners use devices with up-to-date security updates and without malware. Thus, it can be assumed that identities can be securely created and stored on end-user devices and miners can securely use their signing key. In short, we assume that users and miners use secure and trusted systems.
AS2 Confidentiality of Private Keys: Users and miners keep their private keys secret, unless they want to prove the sending or receiving a message. In this case the corresponding one-time identity is disclosed. Note that the private key of the main identity will never be disclosed (A2).
AS3 Secure Cryptographic Primitives: We assume that cryptographic primitives like ciphers, cryptographic hash functions and signature algorithms are practically secure, meaning they cannot be broken in relevant time.
AS4 Global Adversary: We do not consider attackers that can observe or control a large portion of the Internet.

5.3 Security Goals

G1 Confidentiality: The content of the message (R1) as well as the private keys (R2) must be kept confidential (R4).
G2 Anonymity: A key requirement is that parties not involved in a conversation cannot learn anything about the participants nor about the message content itself (R3), except when one of the participants wants to disclose this information.

G3 Non-Repudiation: It is crucial that the author of a message can be identified by the receiving party (R2), and that the author cannot successfully challenge the authorship of a message (R8).
G4 Proof of Existence: The time a message was sent is crucial for certain use cases (R5). Therefore its integrity must be guaranteed.
G5 Immutability: It is important that a message can not be removed or altered after it has been sent (R9) — Neither by the sender nor by anyone else.
G6 Integrity: It must not be possible to tamper with the message or spoof its receiver and sender, without recipients detecting this (R2, R8, R9).
G7 Availability: It must be possible to send and receive messages at any time (R7). Furthermore, old messages need to be available in the network (R4).
G8 Censorship resistance: It is essential that nobody can censor messages without being detected (R9, R10).
G9 Read/Send Confirmation: The sender of a message must be able to prove that it has been sent before a specific point in time (R5). A receiver must be able to prove that she received a message from an individual sender (R8). A sender must be able to prove that a receiver has received an earlier message (R6).
G10 Authentication: The senders' identity must not be forgeable (R2).

5.4 Threats

Analyzing existing approaches and their security analysis targeting relevant problems already yields a baseline for our threat analysis [24–26]. Adapting and extending these analyses for our target domain results in the following threat analysis, also fostering the completeness of the analysis:

T1 Impersonation/Spoofing: An attacker spoofs the senders' identity to impersonate someone else, thus violating G1 and G10.
T2 Message Forgery: An attacker is forging a message, e.g. by encoding random content into transactions, or by modifying an existing transaction. This violates G3, G6, and G10.
T3 Eavesdropping: An attacker eavesdrops on the network communication to learn something, e.g., to get access to the message content or deanonymize a participant. This violates G1 and G2.
T4 Replay Attack: An attacker replays an eavesdropped transaction or extracts its data and encodes it into a new transaction. This violates G3 and G10.
T5 Man-in-the-middle Attack: An attacker actively manipulates the sent transactions or broadcasts faked transactions or blocks. This violates G1, G2, G6, G7, G9, and G10.
T6 Censorship: An attacker deletes transactions from the blockchain or does not forward them, thus violating G4, G7, G8, and G9.
T7 Deanonymization: An attacker deanonymizes a participant by analyzing metadata or by correlation, thus violating G1 and G2.
T8 Denial-of-service Attack: An attacker disrupts the service or attacks individual users, thus violating G7 and G8.

Table 2. Mapping threats to security goals.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
G1	✗	○	✗	○	✗	○	✗	○	✗	○	○
G2	○	✗	✗	○	✗	○	✗	○	✗	○	○
G3	○	✗	○	✗	○	○	○	○	✗	○	○
G4	○	○	○	○	○	✗	○	○	○	○	✗
G5	○	○	○	○	○	○	○	○	✗	✗	✗
G6	○	✗	○	○	✗	○	○	○	○	✗	✗
G7	○	○	○	○	✗	✗	○	✗	○	✗	✗
G8	○	○	○	○	○	✗	○	✗	○	✗	✗
G9	○	○	○	○	✗	✗	○	○	○	✗	○
G10	✗	✗	○	✗	✗	○	○	○	✗	○	○

Table 3. Threat mitigation overview.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
SP1	✓	✓	○	✓	✓	○	○	○	○	○	○
SP2	○	○	✓	○	○	✓	✓	○	○	○	○
SP3	○	○	○	✓	○	✓	✓	✓	○	✓	✓
SP4	○	○	✓	○	○	✓	✓	○	○	○	○
SP5	○	○	✓	○	✓	○	✓	✓	○	○	○
AS1	○	○	○	○	○	○	○	○	○	○	○
AS2	✓	○	○	○	○	○	○	○	○	○	○
AS3	✓	○	○	○	○	○	○	○	✓	○	○
AS4	○	○	○	○	○	○	○	✓	○	○	○

T9 Key Derivation: An attacker calculates the main identity or a different one-time identity based on a published one-time identity. This attack targets G1, G2, G3, G5, and G10.
T10 Manipulating Blocks: An attacker manipulates or deletes blocks from the blockchain with the goal of disrupting the service or deleting sent or received messages, thus violating G5, G6, G7, G8, and G9.
T11 Untrusted Miner: A miner censors transactions, includes manipulated transactions or stops mining, thus violating G4, G5, G6, G7, and G8. Table 2 visualizes the threat to security goals mapping.

5.5 System Properties

SP1 Signature: Every message and reply address is signed by the main identity. The resulting message is signed again by the one-time send identity to prevent replay attacks. Additionally, all transactions have to be signed.
SP2 Encryption: Every message is encrypted after it has been signed twice. The encryption guarantees confidentiality of the message.
SP3 Decentralization: The P2P-network provides a decentralized infrastructure that replicates the blockchain on every node and forwards transactions.
SP4 One-time Identities: Only anonymous one-time identities are used as the receiving or sending address.
SP5 Network-Layer Anonymity: All parties are connected via the Tor network to the Internet and are thus assigned an anonymous IP-address.

5.6 Threat Mitigation

Preventing T1 Impersonation/Spoofing: Spoofing an identity is prevented by SP1. An attacker has to forge a valid signature thus violating AS2 or AS3.

Preventing T2 Message Forgery: Creating a valid signature without the private key would contradict AS3. An attacker could use a published one-time identity to sign a new transaction and include the data from the published transaction. This attack creates a valid transaction, but the consequences depend on the timing of this attack. If the actual communicating parties have exchanged further messages before the attack, the attack will not have any consequences as no one will be listening to this (old) address anymore. If the published transaction belongs to the most recent exchanged message, the recipient will receive the same message twice, but it will not change any of the security guarantees of the previous message. The attacker cannot change the content of the message (SP1) and its content was already published by one of the collocutors.

Preventing T3 Eavesdropping: An attacker eavesdropping on the communication will see transactions with encrypted content (SP2) being sent from one random address to another random address (SP4). As the message is encrypted, the attacker will not learn anything about its content, except the rough size. The use of one-time identities and anonymized Tor exit-node IP-addresses (SP5) prevent insights for an attacker. An exception is a case where one party sends multiple messages in a sequence. In this case, it will be observable that someone received multiple messages, but not that they originated from the same source.

Preventing T4 Replay Attack: Replay attacks are not feasible, as the same transaction will not be included twice into a blockchain (SP3). As soon as some value of the transaction is changed, the transaction signature breaks (SP1, see T2). If the attacker were to extract the encoded data and include it into a new transaction, she would get a valid transaction, but the recipient will notice the attack as soon as she decrypts the message, as the sending identity and the outer signature of the message do not originate from the same entity (SP1). This attack can be used to unnecessarily increase the size of the blockchain.

Preventing T5 Man-in-the-Middle Attack: The success of this attack depends on where the attacker is located. If she is located directly after the exit-node of the Tor network, she could launch all sorts of attacks, like omitting transactions or blocks or sending forged transactions or blocks to the user. Furthermore, she could use her position to start denial-of-service attacks. In all cases, the user will detect that something is wrong and can simply change the exit-node or wait until it is automatically changed (SP5). Omitted transactions are noticed as soon as the following blocks are appended to the blockchain. Omitted blocks are noticed if the chain is broken (one or several blocks are missing) or if the interval where no new block was found is suspiciously large. Manipulated transactions or blocks are noticed because of the broken signature (SP1).

Preventing T6 Censorship: As the interlocutors and the message contents are not known (SP2, SP4), it is not possible to censor messages based on these attributes. An attacker could try to exclude specific IP-addresses or block the service at all. The decentralized structure of the network (SP3) makes

it very hard to censor transactions. Even if one node does not distribute a transaction, other nodes will.

Preventing T7 Deanonymization: As long as the private part of the primary identity is kept confidential (AS2), no connection between one-time identities and the primary identity can be drawn (AS3, SP4). As every message is encrypted (SP2), outsiders can only see one-time identities (SP4) that do not provide any information (SP4). Furthermore, as every node receives all blocks, it is not possible to determine which client received a message (SP3). The IP-address also does not provide any insights, as all nodes are connected over the Tor network (SP5).

Preventing T8 Denial-of-Service (DoS) Attacks: If the attacker controls only some network connections, she cannot harm the network as users can switch their exit node (SP5) and all nodes will forward all valid blocks and transactions (SP3). If the attacker controls the connections of an individual user, this user can be blocked from using the service. An attacker that controls the whole network can block the service, but this attack contradicts AS4.

Preventing T9 Key Derivation: The security of the main identity and the other one-time identities relies on the pre-image resistance feature of the cryptographic hash function (AS3).

Preventing T10 Manipulating Blocks: An attacker would have to delete or manipulate the block on all nodes (SP3), otherwise the network will recover it.

Preventing T11 Untrusted Miner: This attack can be prevented by having a group of miners (SP3), where it is likely that more than half of them are honest. In our scenario, where we likely have only one miner (e.g., the company behind the call for tenders), this attack cannot be prevented, but it will be detected.

All threats except for two — an attacker who controls the Internet connection of the user and an attacker who creates spam messages — are fully mitigated by the design of the proposed solution, as summarized in Table 3.

6 Conclusions and Future Work

In this paper, we have presented a secure and privacy-preserving messaging platform that unifies and improves security and privacy-related features not currently present in any other single messaging solution. The proposed solution is especially designed and suited for business-related scenarios such as bidding processes or tender offers, where data confidentiality is paramount at first, but communication contents are to be divulged at a later point in time. By relying on an adapted Ethereum blockchain and the Clique consensus algorithm, our design scales even beyond the requirements of the targeted use case, since none of the scalability issues typically associated with blockchain-based systems apply. Moreover, advancements made wrt. blockchain scalability also benefit our solution. We therefore expect performance to improve over time.

Our platform relies on two privacy-enhancing pillars which prevent user-profiling, hinder censorship and ensure unlinkable anonymous identities. First, we replace the common centralized service architecture with a distributed architecture based on a blockchain and the Tor network. This allows us to counter a range of issues related to metadata surveillance, eavesdropping, censorship, and control. Second, we separate identities of the parties involved into ephemeral and hidden components, which are derived and associated using a novel address schedule that guarantees their unlinkability for external observers. Using this scheme, all encrypted communication in the network can be traced to the level of ephemeral identities only, which are typically employed only once. When desired, message contents can be published later, including universally verifiable proofs regarding contents and correspondents

We identify two general aspects for further development of our platform. In future work, we first plan to improve platform reliability against spamming and flooding attacks by charging a small amount of currency for every transaction. We also plan to decrease the storage requirements for clients, by allowing them to store only information relevant to them, without losing any security guarantees.

References

1. WhatsApp. One Billion, 1 February 2016. https://blog.whatsapp.com/616/One-billion. Accessed 19 Apr 2017
2. Fiadino, P., Schiavone, M., Casas, P.: Vivisecting WhatsApp in cellular networks: servers, flows, and quality of experience. In: Steiner, M., Barlet-Ros, P., Bonaventure, O. (eds.) TMA 2015. LNCS, vol. 9053, pp. 49–63. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17172-2_4
3. Coull, S.E., Dyer, K.P.: Traffic analysis of encrypted messaging services: Apple imessage and beyond. ACM SIGCOMM Comput. Commun. Rev. **44**(5), 5–11 (2014)
4. Azfar, A., Choo, K.K.R., Liu, L.: A study of ten popular android mobile voip applications: are the communications encrypted? In: 2014 HICSS-47, pp. 4858-4867 (2014). https://doi.org/10.1109/HICSS.2014.596
5. cnet.com: Instant messaging latest trend in e-commerce software (2009). https://www.cnet.com/news/instant-messaging-latest-trend-in-e-commerce-software/
6. Lawton, G.: Instant messaging puts on a business suit. Computer **36**(3), 14–16 (2003)
7. Doyle, S.: Is instant messaging going to replace SMS and e-mail as the medium of choice for direct customer communications? J. Datab. Mark. Customer Strategy Manag. **11**, 17–182 (2003)
8. Cameron, A.F., Webster, J.: Unintended consequences of emerging communication technologies: instant messaging in the workplace. Comput. Hum. Behav. **21**(1), 85–103 (2005). ISSN 0747–5632
9. Gann, R.: Instant messaging for business (2012). https://www.techradar.com/news/world-of-tech/roundup/instant-messaging-for-business-1075434
10. Schneier, B.: NSA doesn't need to spy on your calls to learn your secrets. Wired (2015)
11. Mayer, J., Mutchler, P., Mitchell, J.C.: Evaluating the privacy properties of telephone metadata. Proc. Nat. Acad. Sci. **113**(20), 5536–5541 (2016). https://doi.org/10.1073/pnas.1508081113

12. Schneier, B.: Metadata = surveillance. IEEE Secur. Priv. **12**(2), 84–84 (2014). https://doi.org/10.1109/MSP.2014.28. ISSN 1540-7993
13. Cole, D.: We kill people based on metadata. N. Y. Rev. Books **10**, 2014 (2014)
14. Goldberg, I., OTR Development Team: Off-the record messaging protocol version 3, 6 January 2016. https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html. Accessed 16 Mar 2017
15. Crypviser GmbH: Crypviser - the most secure solution ever (whitepaper), May 2017. https://ico.crypviser.net/static/docs/CrypViserWhitepaper_en.pdf. Accessed 05 Sept 2017
16. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). eprint: https://bitcoin.org/bitcoin.pdf
17. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA, August 2004
18. Buterin, V., et al.: Ethereum white paper (2013). https://github.com/ethereum/wiki/wiki/White-Paper
19. Sziláyi, P.: Clique PoA protocol & Rinkeby PoA testnet #225 (2017). https://github.com/ethereum/EIPs/issues/225
20. Greenwald, G.: NSA collecting phone records of millions of Verizon customers daily. The Guardian **6**(5), 13 (2013)
21. Lyon, J., Wong, M.: Sender id: Authenticating e-mail. RFC Editor, RFC 4406, April 2006
22. Carnegie Mellon University: CERT Division: Spoofed/forged email, September 2017. http://cert.org/historical/tech_tips/email_spoofing.cfm. Accessed 05 Sept 2017
23. Brown, D.R.L.: Standards for Efficient Cryptography 2 (SEC 2), January 2010. http://www.secg.org/sec2-v2.pdf. Accessed 27 July 2018
24. Petrlic, R., Sorge, C.: Instant messaging. Datenschutz, pp. 97–108. Springer, Wiesbaden (2017). https://doi.org/10.1007/978-3-658-16839-1_8
25. Schrittwieser, S., et al.: Guess who's texting you. Evaluating the Security of Smartphone Messaging Applications, SBA Research gGmbH (2012)
26. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. In: 2017 IEEE EuroS&P, April 2017, pp. 451–466. https://doi.org/10.1109/EuroSP.2017.27